TP 3

1 En finir avec le passé

Finir avec soin les 2 TPs précédents.

2 Construction d'une classe d'objets fonctions

Construisez une classe dont les instances seront des générateurs d'entiers, multiples d'une valeur donnée à l'initialisation de l'instance. Les valeurs fournies par un générateur vont en augmentant au fil des sollicitations (invocation de l'opérateur () sur l'objet). Par exemple un générateur de multiples de 5 renverra les valeurs 5, puis 10, puis 15, etc.

3 De C++ à Java ou Vice Versa (le retour)

La correction fournie à cet exercice du TP1 fait apparaître deux implémentations `C++` de la classe `Image`. L'une d'entre elles est efficace et économique en mémoire (`Image` correspondant à des Pixels contigus en mémoire), l'autre non (`Image2` correspondant à des Pixels accessibles par indirection comme en Java) !

Munir les classes de ces deux implémentations de mécanismes de construction et d'affectation par déplacement.

Imaginons à présent que l'on veuille que les images de type `Image2` stockent des `Pixel`s polymorphes (avec deux spécialisations possibles suivant qu'il s'agit d'un Pixel en couleur ou en niveau de gris). Dans ce cas, il faudra que `Pixel` devienne la racine d'une hiérarchie de classes avec des fonctions membres virtuelles. Il faudra également modifier la classe `Image2` avec deux initialisations possibles suivant que les pointeurs sur `Pixels` pointent sur des `PixelsCouleurs` ou des `PixelsNiveauDeGris`. Il n'y a donc pas une hiérarchie de classe `Image2`, mais des `Pixel`s de nature différente suivant le constructeur appelé. Faites les modifications requises. Que constatez-vous en terme de surcoût en temps d'exécution et de mémoire ? Le langage `C++` est-il encore compétitif par rapport à `Java` ?